**PATENT**

**Amendments to the Specification:**

Please replace the paragraph beginning at page 3, line 18 with the following amended paragraph:

The final part of an application in accordance with a preferred embodiment is a set of work objects. Each of these objects is modeled with ~~a java class~~ a JAVA™ class. (The JAVA™ programming language is a trademark of Sun Microsystems.) A work object describes a unit of work in the system, perhaps analyzing a mail message for some string or sending a message to a project manager. Work objects are created by your application anytime that you desire to perform a complex computation. This structure is a bit more work for the application developer, but allows the platform to use queues internally for scheduling work to be done and insuring that work is completed in a timely fashion. Features including redundancy and load balancing are introduced transparently to the application developer utilizing the work objects.

Please replace the paragraph beginning at page 7, line 3 with the following amended paragraph:

A preferred embodiment provides a base of functionality that application developers can use to add value to email messages. In addition, the system provides a set of reusable, programming abstractions that conceal the details of delivering application functionality in email. This functionality allows application developers to focus on the particular business process they are trying to automate without being concerned about the underlying processing. A set of work objects are each modeled with ~~a java class~~ a JAVA class. Each work object describes a unit of work in the system, perhaps analyzing a mail message for some string or sending a message to a project manager. Work objects are created to perform a set of complex computations. This structure is a bit more work for the application developer, but allows the platform to use queues internally for scheduling work to be done and insuring that work is completed in a timely fashion. Features including redundancy and load balancing are introduced transparently to the application developer utilizing the work objects.

Please replace the paragraph beginning at page 13, line 19 with the following amended paragraph:

Property groups are a set of property names and property value types grouped together for a program's use. In one aspect, they may be encoded ~~in a Java class~~ in a JAVA class - "in code" - although the idea works with other encodings. For example, a common property group is the set of properties that structure information for browsing. This group might be written something like this:

- Browser.name: java.lang.String
- Browser.size: java.lang.Integer
- Browser.creation: java.util.Date

Please replace the paragraph beginning at page 16, line 8 with the following amended paragraph:

Figure **5** is an illustration of an exemplary unstructured table **500** in accordance with an embodiment of the present invention. Because some, perhaps even many, properties will not be participants in any property group, a table may be stored that has these "unstructured" properties. Such a table may be referred to as an unstructured table. In an unstructured table, each row is roughly a property name-value pair. In the exemplary unstructured table **500** illustrated in Figure **5** includes four columns: a Document ID **502** column, a Property Name column **504**, a Property Value Column **506**, and a Hash column **508**. In this table **500**, two unstructured properties **510**, **512** are on document 209, and one **514** is on document 10472. The values of unstructured properties may be stored as serialized ~~Java objects~~ JAVA objects in SQL Blobs in the column Property Value. The Hash column **508** of this table **500** may be used to make equality queries fast. In one aspect of the present invention, the Hash value may be determined by calling the "hashcode()" method on the ~~java object~~ JAVA object that is the value of the property. Since the database cannot interpret the serialized object in the value column when evaluating a query, one can use the hash value column (that is understood by the database) to give a conservative estimate of equality. This may require that some false positives be removed after the database does an evaluation involving data in this table.

Please replace the paragraph beginning at page 21, line 12 with the following amended paragraph:

Sun Microsystems's ~~Java language~~ JAVA™ programming language solves many of the client-side problems by:

- Improving performance on the client side;
- Enabling the creation of dynamic, real-time Web applications; and
- Providing the ability to create a wide variety of user interface components.

Please replace the paragraph beginning at page 21. line 17 with the following amended paragraph:

~~With Java~~With JAVA, developers can create robust User Interface (UI) components. Custom "widgets" (e.g., real-time stock tickers, animated icons, etc.) can be created, and client-side performance is improved. Unlike HTML, ~~Java supports~~ JAVA supports the notion of client-side validation, offloading appropriate processing onto the client for improved performance. Dynamic, real-time Web pages can be created. Using the above-mentioned custom UI components, dynamic Web pages can also be created.

Please replace the paragraph beginning at page 21, line 24 extending to page 22 line 5 with the following amended paragraph:

Sun's ~~Java language~~ JAVA™ programming language has emerged as an industry-recognized language for "programming the Internet." Sun ~~defines Java~~ defines JAVA as: "a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, dynamic, buzzword-compliant, general-purpose programming language. ~~Java supports~~ JAVA supports programming for the Internet in the form of platform-independent ~~Java applets~~ JAVA applets." ~~Java applets~~ JAVA applets are small, specialized applications that comply with Sun's ~~Java Application~~ JAVA Application Programming Interface (API) allowing developers to add "interactive content" to Web documents (e.g., simple animations, page adornments, basic games, etc.). Applets execute within a ~~Java compatible~~ JAVA-compatible browser (e.g., Netscape Navigator™ browser) by copying code from the server to client. From a language standpoint, ~~Java's core~~ JAVA's core feature set is based on C++. Sun's ~~Java~~

~~literature.~~ JAVA literature states that ~~Java is~~ JAVA is basically, "C++ with extensions from Objective C for more dynamic method resolution."

Please replace the paragraph beginning at page 26, line 15 with the following amended paragraph:

Figure **10** is an architecture diagram that shows the orientation of a preferred embodiment **1000** with respect to other platform and application layers. The ThinkDoc platform **1004** is a technology layer, as shown in Figure **10**, that sits above the ~~java platform~~ JAVA platform **1006** and below the ThinkDoc applications **1002**. The ThinkDoc platform **1004** provides a base of functionality that application developers can use to add value to email messages. Further, the platform **1004** provides a set of reusable, programming abstractions that conceal the details of delivering application functionality in email. This allows application developers to focus on the particular business process they are trying to automate with being concerned about the underlying processing.

Please replace the paragraph beginning at page 26, line 26 with the following amended paragraph:

The platform is preferably written entirely ~~in Java~~ in the JAVA programming language and thus should be portable to any computing environment that supports the ~~Java 2 standard edition~~ JAVA 2 Standard Edition. It has been tested on Windows™ (98, NT4, and 2000) as well as SOLARIS ™ 7. JAVA can ~~Solaris 7. Java can~~ be used for all development.

Please replace the paragraph beginning at page 27, line 2 with the following amended paragraph:

Developers building ThinkDoc applications can use the ThinkDoc toolkit to develop their application. This toolkit is part of the platform and is an object-oriented class library that provides the abstractions mentioned above. It provides support, for example, for describing incoming email messages that an application wants to receive and outgoing messages that the application wishes to send. The ~~(Java)~~ (JAVA) interfaces supported by this toolkit are intimately linked with the platform; for example, the platform's notion of a

message template that describes the types of messages that an application might send is described by the interface MsgTemplate. To send a message using the platform, an application writer must use the MsgTemplate interface, although they can specialize this for their purposes.

Please replace the paragraph beginning at page 31, line 24 and extending to page 32, line 4 with the following amended paragraph:

The final part of an application is a set of work objects. Each of these objects is modeled with ~~a java class~~ a JAVA class. A work object describes a unit of work in the system, perhaps analyzing a mail message for some string or sending a message to a project manager. Work objects are created by an application anytime that it is desired to perform a complex computation. This structure is a bit more work for the application developer, but allows the platform to use queues internally for scheduling work to be done and insuring that work is completed in a timely fashion. Preferably, the platform can introduce redundancy and load balancing transparently to the application developer because of these Work objects.

Please replace the paragraph beginning at page 40, line 11 with the following amended paragraph:

Referring to Figure **15**, when MD1 **1404** is asked if it matches the presented to line, someone-57@somehost.thinkdoc.com, it will be asked to return an object. This object is represented as a "Blob Of Data" BOD or "any old object" **1502**. This can be ~~any java~~ any JAVA Object. This object is not interpreted by the ThinkDoc infrastructure itself, it is simply held by the infrastructure to be returned to MD1 **1404** later.

**Amendments to the Abstract:**

Please cancel the current Abstract and replace it with the following new Abstract:

ABSTRACT

A method, system and computer program for operating an application server using email messages uses the address format of the email address of an incoming email message to invoke an application to process the content of an incoming message. Each application available on the server defines acceptable address formats of the email addresses of messages that it will accept for processing. An application may generate a reply to an incoming email message, or may generate an event, to advance processing of a task. The application server may also determine a recipient email address of an actual recipient of the incoming e-mail, and transmit the incoming e-mail message to the recipient email address. In this case, the application server functions by interposing itself in the email channel between two recipients.